Configuration Management

Shanika Karunasekera

Department of Computing and Information Systems

The University of Melbourne

karus @unimelb.edu.au

2019 – Semester 1

Lecture 11

Learning Outcomes

1. Understand the fundamentals of quality management

2. Understand the quality management process

3. Understand the following quality management activities:

   – Quality Assurance

   – Quality Planning

   – Quality Control and Monitoring

Software Quality

External Characteristics → End User's Perspective

Internal Characteristics → Developer's Perspective

Cost of Quality

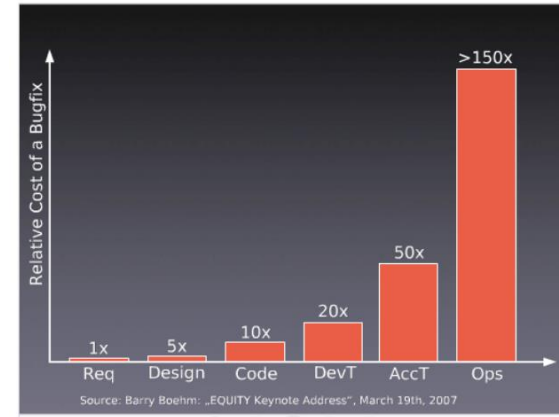
Source: Barry Boehm: „EQUITY Keynote Address", March 19th, 2007

Quality Management Process

Quality Assurance → Quality Planning → Quality Control

Quality Planning

Quality Goals
Verification vs Validation

Verification & Validation Activities

Technical Reviews
Business Reviews
Management Reviews
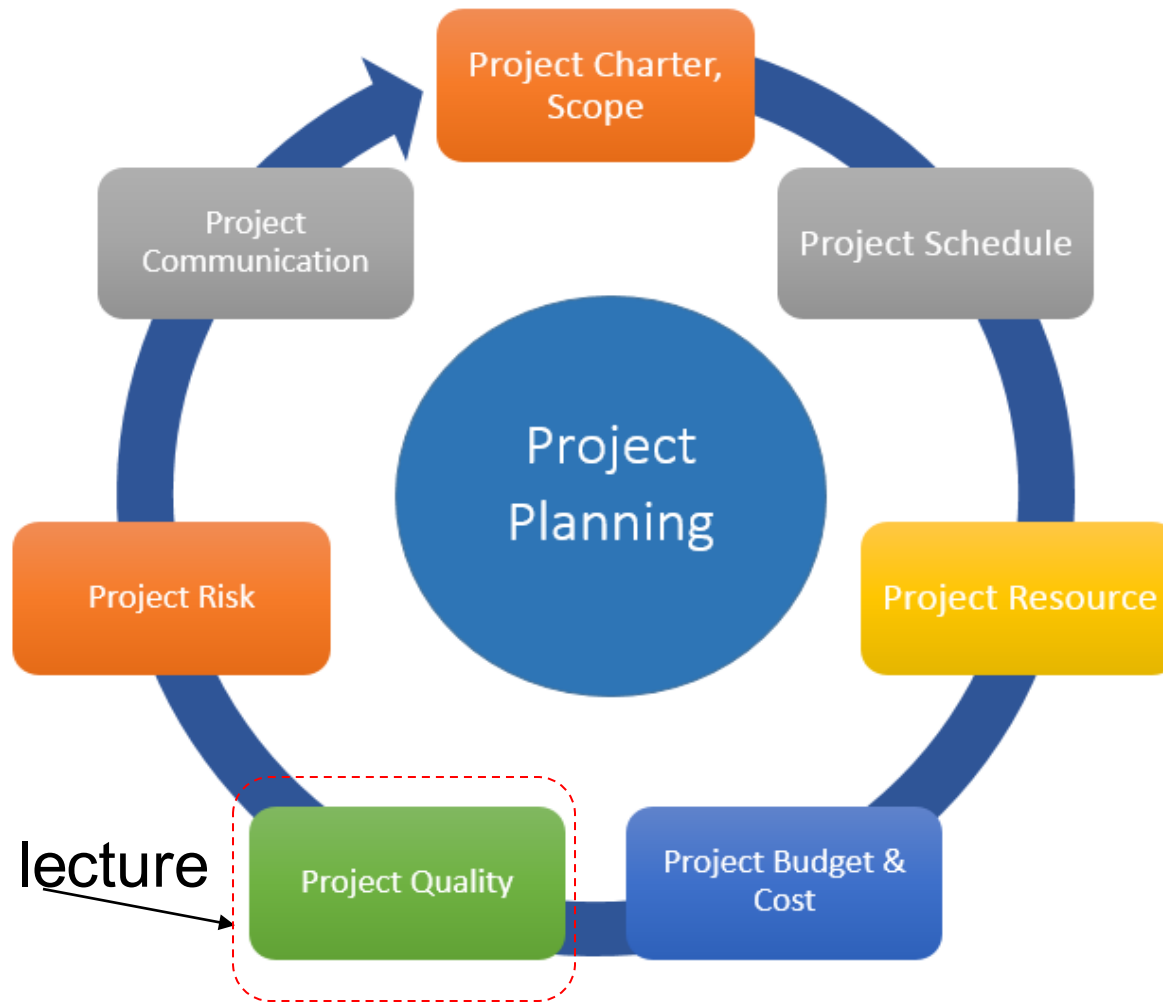**Audits**

Which one of the following is an external quality attribute?

Testability **A**

Portability **B**

Usability **C**

Modularity **D**

Reusability **E**

Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com/app**

Today's lecture → Project Quality

Semester Structure

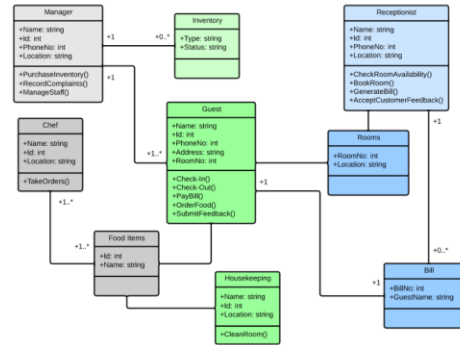| Week #	Lecture Date	Lecture Law G15 – Thursday 9.00am to 11.00am	Assignment
1	07/03/19	Subject Introduction, Introduction to Projects and Project Management,	
2	14/03/19	Project Management Plan & SDLC's	Assignment 1 Spec available on LMS 15/3
3	21/03/19	Individuals, Motivation and Teams	
4	28/03/19	Stakeholder Management Communication Management	Assignment 2 available & Groups created during the workshops / tutorials – attendance mandatory
5	04/04/19	Project Planning and Scheduling Assignment 1 & 2 open forum / discussion	Assignment 1 (Individual) due Fri 5/4 @ 11.59 pm
6	11/04/19	Cost Estimation	
7	18/04/19	Risk Management	
	25/04/19	*Non Teaching Week – Mid semester break*	Assignment 2 (Part 1) due Wed 24/4 @ 11.59 pm
8	02/05/19	Quality Management	
9	09/05/19	Ethics, Outsourcing & Procurement	Assignment 2 (Part 2) due Sat 11/5 @ 11.59 pm
10	16/05/19	Guest Lecture	Assignment 2 (Part 3) due Sat 18/5 @ 11.59 pm
11	23/05/19	Configuration Management	Assignment 2 (Final) due Sat 25/5 @ 11.59 pm
12	30/05/19	Subject Revision and Exam Prep	Assignment 2 Project Demonstration during tutorials

1. Understand the role of configuration management

2. Understand the configuration management process

3. Understand the tasks associated with configuration management

1. Understand the role of configuration management

2. Understand the configuration management process

3. Understand the tasks associated with configuration management

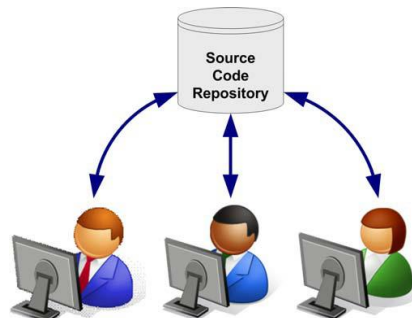- Software projects generate a large number of different types of artefacts – e.g.:

Use-case diagrams

Class diagrams

Software Requirements Specification

Software Architecture Document

Software Design Document

Software Test Plan

Test Cases

Source Code

- There are *dependencies* between all of these artefacts.
  - For example, a code module may depend on a design element such as a class diagram or state chart, as well as on a design element such as a design class diagram. In turn these may depend on a combination of textual requirements, use-cases and analysis classes.
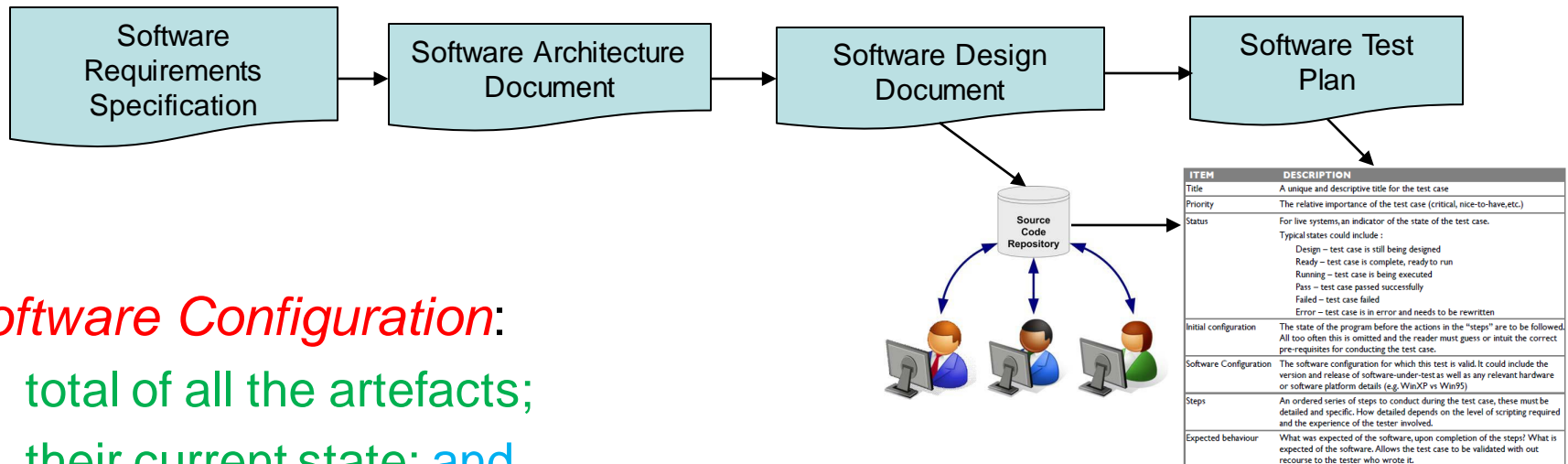
- *Software Configuration*:
  - total of all the artefacts;
  - their current state; and
  - the dependencies between them.

THE UNIVERSITY OF
MELBOURNE

The problem is change!

- If we make a change to an artefact, it may impact all of that artefact's dependencies

- If we are not careful then changes to artefacts may leave the configuration in an inconsistent state

  – For example, a change to the requirement will have an impact on the system design and all of the code modules that depend on the design. Also, the test plan, test cases and testing scripts for the code will also be impacted. *The danger is that we may change one module without changing one of its dependent modules leaving the configuration inconsistent.*

- The aim of configuration management is to manage change properly without losing overall consistency through:
  - establishing processes;
  - setting up repositories; and
  - using other appropriate tools and techniques

- Configuration Management (CM) addresses the following:
  - How do we manage requests for change?
  - What and where are the software components?
  - What is the status of each software component?
  - How does a change to one component affect others?
  - How do we resolve conflicting changes?
  - How do we maintain multiple versions?
  - How do we keep the system up to date?

1. Understand the role of configuration management

2. Understand the configuration management process

3. Understand the tasks associated with configuration management

CM Aims:

1. To identify all items that collectively will make up the configuration

2. To manage changes to one or more of these items so that the collection remains consistent

3. To manage different versions of the product

4. To assure software quality as the configuration evolves over time

Reporting

Configuration Auditing

Change Control

Version Control

Identification

SCIs
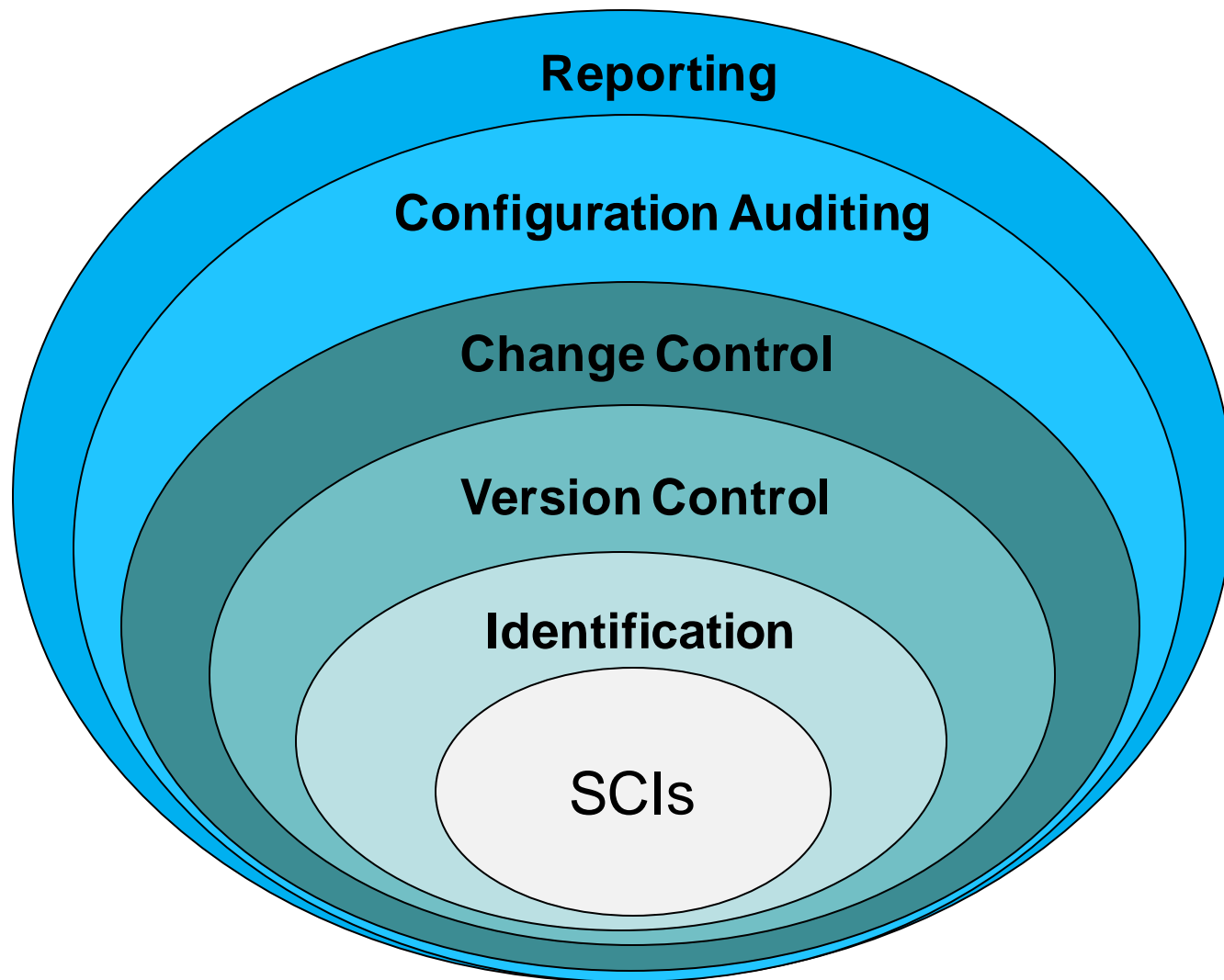
- Identification
  - the configuration items necessary for the project are identified
- Version control
  - processes and tools are chosen to manage the different versions of configuration items as they are developed
- Change control
  - changes that affect more than just one configuration item are managed
- Configuration auditing
  - the consistency of the configuration is checked
- Configuration reporting
  - the status of configuration items is reported

1. Understand the role of configuration management

2. Understand the configuration management process

3. Understand the tasks associated with configuration management

- The set of artefacts that require configuration management are called the *configuration items*

- *Configuration Items:*
  - Basic
  - Aggregate
  - Derived
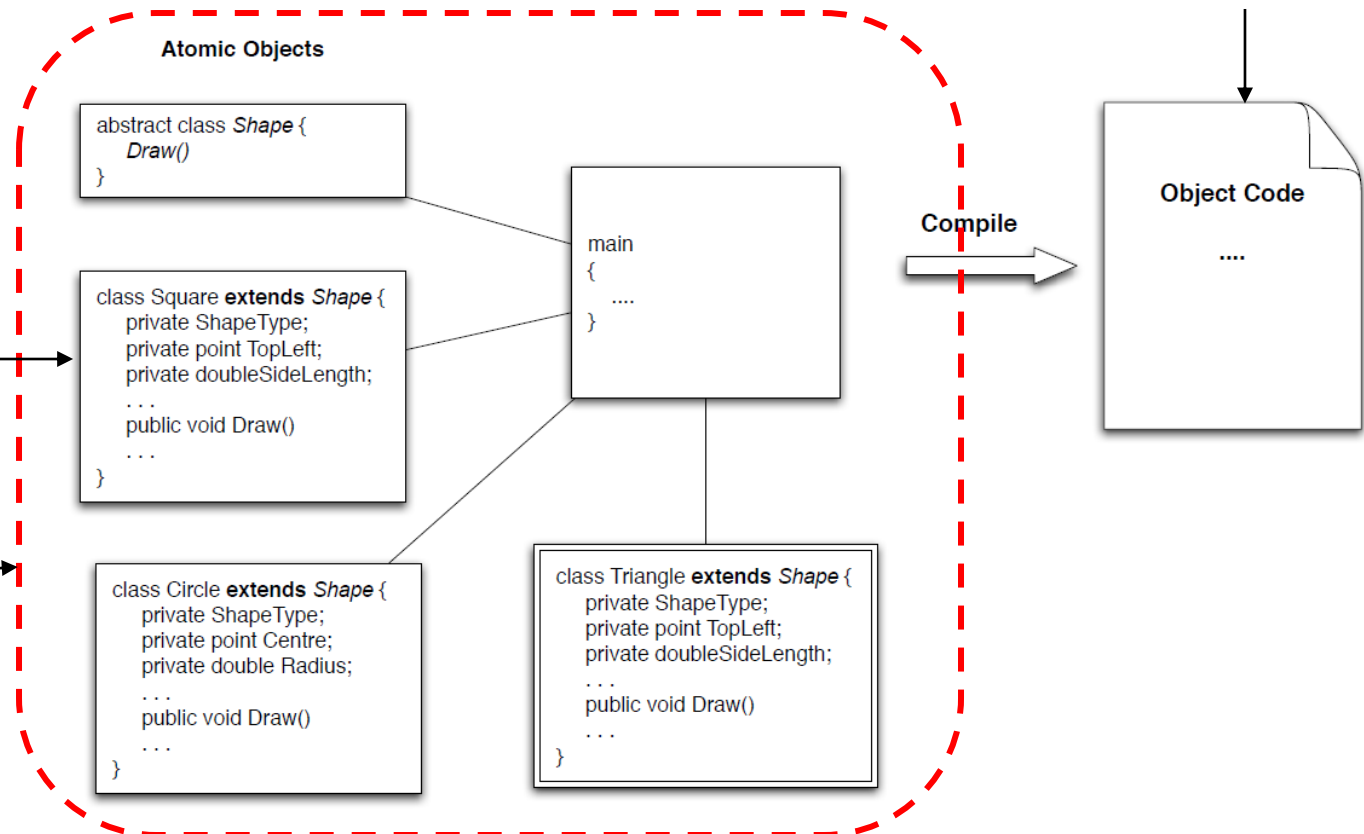
- A typical list of configuration items
  - requirements specifications, requirements models, sections of the requirements specification, and individual requirements
  - use-cases, user stories
  - design models, design documents, design elements, and class designs
  - source code modules
  - object code modules
  - release modules
  - software tools
  - test drivers and stubs, and test scripts
  - documents or sections of documents associated with the project

- Requirements for a version control system:
  1. A *repository* for storing configuration items
  2. A *version management function* that allows software engineers to create and track versions, and roll the system back to previous versions if necessary – e.g. git, svn, cvs
  3. A *make-like facility* that allows engineers to collect all of the configuration objects for a particular target together and to build that target – *e.g. Apache Maven, Apache Ant, make (unix, linux)*

- SCM information is maintained in a *repository* or *configuration database*
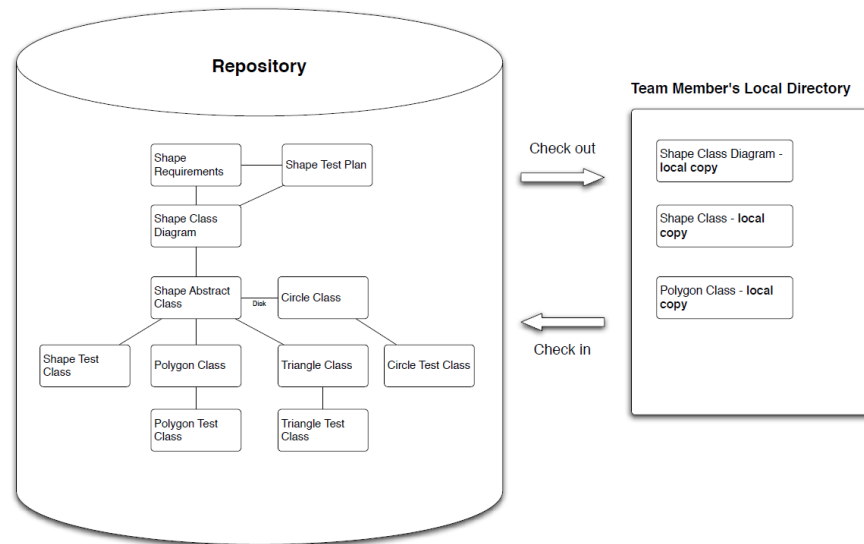

Version*:* An instance of a model, document, code, or other configuration item which is functionally distinct in some way from other system instances.
Variant: An instance of a system which is functionally identical but non-functionally distinct from other instances of a system.
Release: An instance of a system which is distributed to users outside of the development team.

- *Derivation History:*
  - This is a record of changes applied to a configuration object

- Each change should record contain:
  - the change made
  - the rationale for the change
  - who made the change
  - when it was implemented

- A common method of tracking versions in a repository is through *version numbering*
  - Version numbers could have meanings – for example a reviewed version of a document (major versions) vs un-reviewed changes
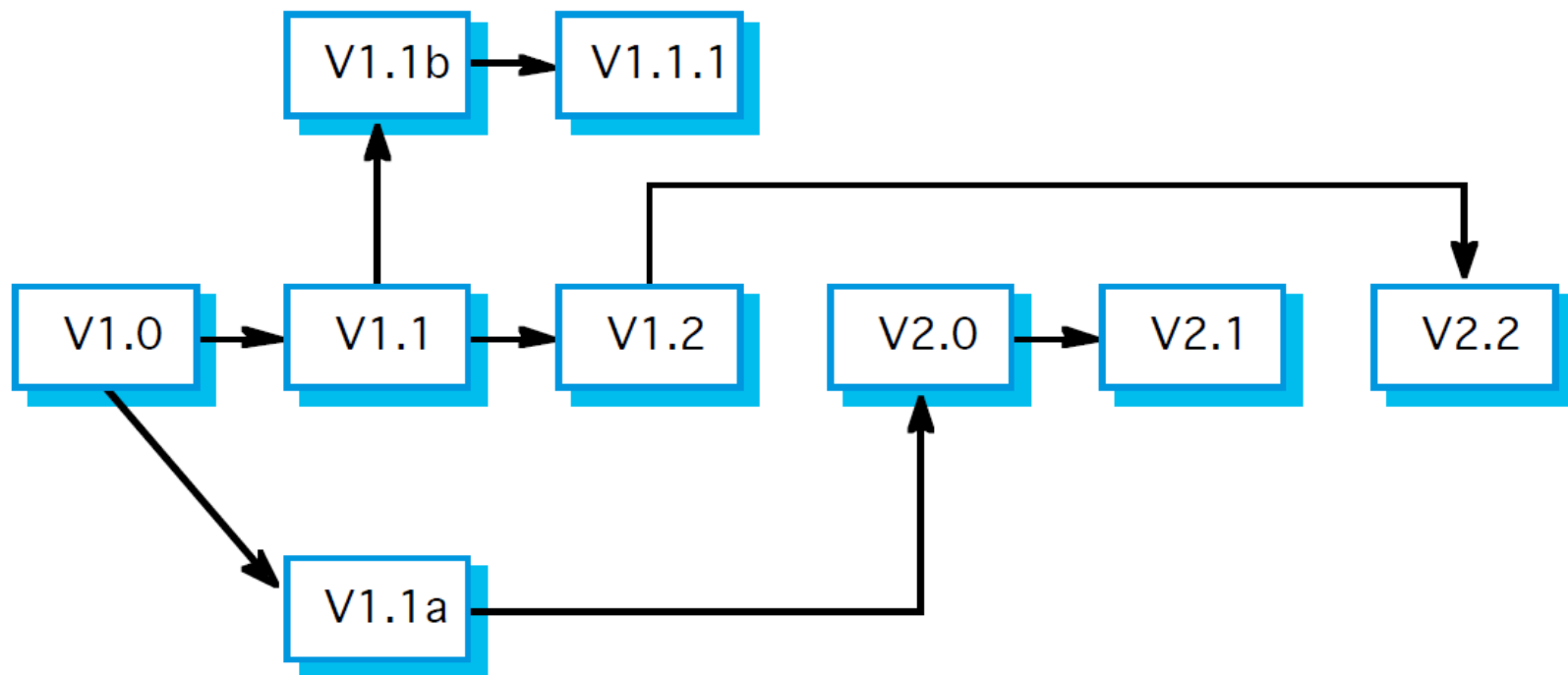
A derivation structure for a project using version numbering to mark branches and merges

- *Change Management Plan*
  - A part of an overall configuration management plan to specifically control these changes to the configuration
  - Changes must be made in a way that allows everyone on the project team to find out:
    - exactly what changes need to be made
    - what they need to do to affect the change
    - why the change is being made
    - how it will impact them

    More importantly, in distributed control structures, some changes may need to be carefully negotiated so that everyone understands the need for the change and supports it

Change Control

| Initiate the Change | → | Evaluate the Change | → | Make the Change |

Element	Impact on the Process
Initiate the Change	Why is the change being made? What information will be needed to evaluate the change? How will the change be evaluated?
Evaluate the Change	How will the change affect the configuration? Which artefacts need to change and what are their dependencies? What are the benefits of the change? What are the costs of the change? Do the benefits of the change outweigh the costs of the change? Who will be impacted by the change?
Making the Change	Who will put the change into effect? How will the change be managed? How will other people working on the project understand the change? How will they be notified of the change? How will people working on the project know when the change is completed?

- ***Baseline***

  - A baseline is an artefact that is *stable*

  - It has been formally reviewed and agreed upon, that is now ready for future development

  - It can only be changed through a *formal change management procedure*

- *Configuration audits:*
  - complement the other configuration management activities by assuring that what is in the repository is actually consistent and that all of the changes have been made properly

Have the changes requested and approved been made?	Have any additional changes other than required by a request been made?
Did the configuration objects that were changed pass their quality assurance tests?	Do the objects in the configuration meet the required external standards?
Do the attributes of the configuration item match the change?	Does every configuration item have appropriate change logs?

Typical questions for a configuration audit

- *Status Reporting*
  - Is a common way for large projects to keep track of the status of the repository
  - The idea is to review the configuration objects for consistency with other configuration objects, to find any omissions or to look for potential side effects
  - Status reporting can take many forms, but most commonly the aim is to report on the status of the configuration items of interest and the baselines that have been achieved
    - For example, we may have a design element that is in one of the states: not-initiated, initial-work, modified, approved, baselined – the status report can compare the state with what is in the project schedule

1. Understand the role of configuration management

2. Understand the configuration management process

3. Understand the tasks associated with configuration management

Which one of the items below is not a part of a change record?

the change made

the repository it is stored in

who made the change

the rationale for the change

when it was implemented

Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com**/app